# A scalable system for community discovery in Twitter during Hurricane Sandy

Yin Huang
*Computer Science and Electrical Engineering Department*
*University of Maryland, Baltimore County*
*Email: yhuang9@umbc.edu*
*IBM Canada Software Laboratory, CAS Research*
*Markham, Canada*

Han Dong, Yelena Yesha, and Shujia Zhou
*Computer Science and Electrical Engineering Department*
*University of Maryland, Baltimore County*
*Email:{han6, yeyesha, szhou}@umbc.edu*

***Abstract*—The wide use of microbloggers such as Twitter offers a valuable and reliable source of information during natural disasters. The big volume of Twitter data calls for a scalable data management system whereas the semi-structured data analysis requires full-text searching function. As a result, it becomes challenging yet essential for disaster response agencies to take full advantage of social media data for decision making in a near-real-time fashion. In this work, we use Lucene to empower HBase with full-text searching ability to build a scalable social media data analytics system for observing and analyzing human behaviors during the Hurricane Sandy disaster. Experiments show the scalability and efficiency of the system. Furthermore, the discovery of communities has the benefit of identifying influential users and tracking the topical changes as the disaster unfolds. We develop a novel approach to discover communities in Twitter by applying spectral clustering algorithm to retweet graph. The topics and influential users of each community are also analyzed and demonstrated using Latent Semantic Indexing (LSI).**

***Keywords*-Hadoop; HBase; Lucene; Twitter; Hurricane Sandy**

## I. Introduction

In the event of natural disasters, social media or networking services usually take precedent as the main form of communication for emergencies or evacuation news. As evidenced in the Great East Japan Earthquake of March 2011, web-enabled smartphones served as the primary information channel for communication [1]. Hurricane Sandy was considered to be the second-costliest hurricane in United States, which affected much of the east coast from October 22, 2012 to October 30, 2012. Social media can play a significant role in disseminating vital information and serve as a pool of information for understanding public sentiment during natural disasters [2]. Our approach differs from [2] in that we use HBase to store the data in a distributed data management system rather than in a centralized way.

Typical examples of social media include Twitter, Google+ and Facebook etc. In this work, we focus on the Twitter as it has a significantly large user base as well as an API for extracting data. "Big social data" has become a global phenomenon and has grown in complexity in terms of volume (terabyte to petabyte), variety (structured and un-structured), and velocity (high speed) in nature. Computational and storage requirements of applications such as Business Intelligence (BI), and social networking analytics have led to the development of horizontally scalable, distributed non-relational data stores like "Not only SQL" (NoSQL) databases [3].

HBase, one member of NoSQL databases, is built on top of Hadoop and consists of the following two components: Hadoop Distributed File System (HDFS), an open source implementation of Google File System (GFS), and a powerful parallel processing framework in MapReduce. While the benefits of HBase involve availability, scalability and load balancing, it has less querying capability and often weaker consistency guarantees [4]. For example, HBase does not support full-text searching; current implementation of HBase only offers the rowkey-based indexing. As a result, performance could be degraded significantly when querying on column fields that are not indexed by the rowkey. In Twitter, we care about the *author*, *time*, *location* and *content* of the tweets for analysis. The former three attributes are well-defined in our HBase table schema, while the unstructured *content* requires full-text searching power.

Lucene [5] is an open-source Apache sub-project, which provides mature Java-based indexing and searching technology. Lucene has been applied into numerous successful search engines such as Apache Nutch [6], Solr [7], ElasticSearch [8] etc. Lucene uses inverted indexes and is comparable with sequential indexes. Sequential indexes map from documents to words whereas inverted indexes maps from word to document. Accordingly, given a keyword, one can easily find all the related documents containing the keyword simply by looking up Lucene index.

Discovering communities in the context of social media during disasters can help tracking topical changes, identifying influential factor, and detecting the evolvement of communities. Such information benefits relevant agencies to respond in a real-time way. The discovery of communities resembles graph clustering problems. We are looking for graph cuts so that links with members in the same community are dense while links among members of different communities are sparse. Nodes are linked if there is a relationship between them, weighted edges represent their similarities. Typical metrics for constructing this similarity graph include social connections, tweet content similarities, mentions, and descriptions content similarity [9].

The motivation of our work comes from the spatial and temporal characteristics of disasters. Social media data analytics involve multidimensional queries, which specify a particular location, time and content to reach a finer understanding of human behaviors in the face of disasters. We build Lucene index layer on top of HBase to retrieve relevant data in a near-real-time manner for analysis. Such a prototype can be employed in similar social media analytics that require multidimensional analysis. However, we are currently not aiming at stream processing of Tweets with a system like Apache Storm [10].

The rest of the paper is organized as follows. Section II is related work. An overview of our proposed scalable data analytics system is described in Section III. Section IV presents details about building Lucene index on top of HBase. Section V demonstrates the scalability of our system. Section VI focuses on community discovery in Hurricane Sandy using spectral clustering followed by discussion and conclusion in Section VII and Section VIII.

## II. Related work

Existing solutions to use Lucene to support full-text searching for HBase include HBasene [11], Lily [12], and IHBase [13]. HBasene is a distributed system that uses HBase as the backing store for the TF-IDF representation. HBasene aims at combining HBase with Lucene by modifying HBase APIs. Lily is a data management system combining planet-size data storage, indexing and search with online, real-time usage tracking, audience analytics and content recommendations. It relies on Solr as the index engine and stores data in HBase. IHBase stores indexes in memory, which might not be suitable for large table sets. Gao uses MapReduce to build Lucene index on HBase and stores indices in a HBase table [14]. In this paper, we use MapReduce to directly build Lucene index for HBase tables and store the indices on HDFS where queries are searched against the indices copied to local file system from HDFS.

In literature, five broad classes of community detection are considered: (a) cohesive subgraph discovery [15], (b) vertex clustering, (c) community quality optimization [16], (d) divisive [17], and (e) model-based [18]. In this paper, we take the vertex clustering approach, which is a typical means of casting a graph vertex clustering problem to one that can be solved by conventional data clustering methods [19]. We first construct a similarity matrix by retweets, then apply spectral clustering to the matrix to discover different communities. The reason for using retweet as the similarity relationship lies in the fact that retweets inherit the dynamic and temporal features of a disaster; traditional metrics such as social connections fail to capture such characteristics. In addition, retweet also reflects the trust between two people.

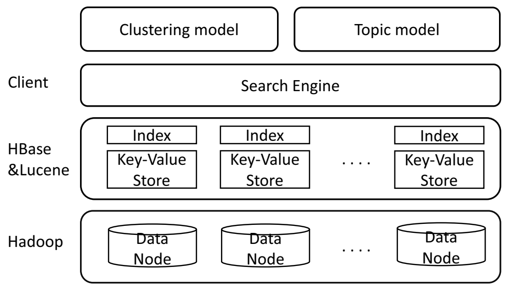

Figure 1. Architecture of the scalable data analytics system

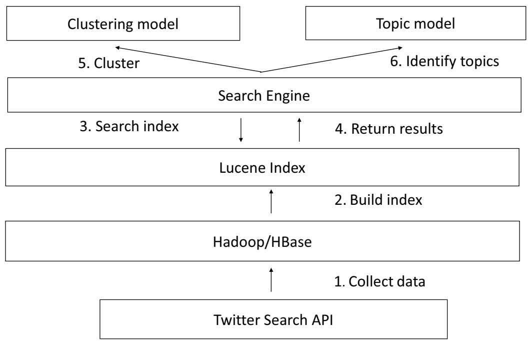

Figure 2. Data analytics flow

## III. System Overview

Figure 1 shows the architecture of our scalable data analytics system while Figure 2 shows the data analytics flow. The first layer is the twitter search API followed by data storage layer where twitter data are collected and stored in HDFS. We import the data collected using twitter search API into one HBase table called *Twitter*. The third layer is the index layer where Lucene provides the full-text searching capacity to HBase. Raw HBase and Hadoop APIs are used to build Lucene index for *Twitter* using MapReduce. The fourth layer is the search engine, where the client can issue queries against the *Twitter* table in multiple dimensions. Multidimensional queries such as "location:NYC and time: Oct 28 to 31 and content: evacuate" are answered within milliseconds.

To evaluate the performance of our system, the following four components should be taken into consideration: (1) indexing, (2) searching, (3) clustering, (4) Latent Semantic Indexing (LSI). To improve the indexing time, we rely on MapReduce to build Lucene index for *Twitter*. The end-user will then issue multidimensional queries to fetch data at a particular location, in a particular time and with particular keywords. The shorter the query response time, the better the performance. The query results serve as the input for clustering. After we identify the communities, LSI is applied to each community to identify the topic. As New York City (NYC) suffered most during Hurricane Sandy on October

29, 2012, we focus on analyzing twitter communities in NYC to understand public reactions, the evolvement of communities, and identify the most influential users.

## IV. Lucene index for HBase

HBase is similar to BigTable [20], which is a distributed, fault-tolerant, highly-scalable, NoSQL database for large volumes of heterogeneous data. The data model in HBase is similar to the one used by BigTable, which organizes data in tables, rows and columns. Each table is composed of millions of rows; each row, identified by a unique rowkey, consists of various number of column families, where each column family may have arbitrary number of columns. Table I illustrates the *SandyTwitter* data schema. *SandyTwitter* is built by searching *Twitter* table for retweets related to Hurricane Sandy.

Table I
SandyTwitter table schema. Rowkey is composed by author, location and time. Cf is the only column family. Target is the original author of a retweet

| | | | | | |
|---|---|---|---|---|---|
| Rowkey | Cf:author | Cf:location | Cf:time | Cf:content | Cf:target |

Physically, HBase tables are stored in HDFS files in a distributed manner. These tables are automatically partitioned horizontally by HBase into regions. Each region comprises a subset of the table. In addition, HDFS replicates the HBase tables to prevent the data loss due to node failures.

Logically, rowkey is used to address all columns in one single row, and it is sorted to speed up searching; column key is the combination of a column family name and a column qualifier. Table name, rowkey, and column key together with the timestamp define a cell in the table. In order to access the data, the rowkey must be provided to retrieve the cell.

Lucene indexes are composed by searchable entities represented as documents including field and value pairs. In addition, Lucene supports multiple query features, like wild card query, range query, and customized parsers. To speed up the indexing of HBase tables, we utilize MapReduce to parallelize the computation. Indexing an HBase table is an embarrassingly parallel task. Each mapper reads rows from the table stored in the corresponding regionserver. Each row from HBase table is indexed into a lucene document with each column stored in the corresponding lucene field. As soon as a mapper completes its job, the index folder is copied to a directory in HDFS.

## V. Experimental results of building Lucene index

We designed a set of experiments to test the scalability and efficiency of the system. The 4 sets of clusters using Amazon Elastic Compute Cloud (Amazon EC2) were built. Each cluster consisted of 1, 4, 8, and 12 nodes separately, using Amazon EC2 general purpose m1.medium instance which has 64 bit Intel Xeon processor, 1 vCPU, 2 ECUs, 3.75GB memory, and 160GB HDD. These instances launched into the same cluster placement group were placed into a non-blocking 10 Gigabit Ethernet network. We installed 64-bit Ubuntu Server 12.04.3 LTS with Oracle Java 7, Hadoop 1.0.4, HBase 0.94.0 and Zookeeper 3.4.3.

We have collected 50GB twitter data. However, it took about 2.5 hours to import 6GB data into Amazon EC2 clusters. We decided to test the performance with this 6GB *twitter* table which had the same schema except the target shown in table I. Rowkey was composed of column author, location, and time; such design could overcome the hotspot problem caused by unevenly distributing the data among regionservers.

Table II shows the cluster configurations and running time

Table II
Cluster configurations, Hadoop setup time and total time to index 6GB data

| Group | Number of nodes | Data nodes | RAM | Hadoop setup time | Total time |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 3.75GB | 6sec | 45min 52sec |
| 2 | 4 | 2 | 3.75GB | 15sec | 23min 11sec |
| 3 | 8 | 6 | 3.75GB | 50sec | 11min 32sec |
| 4 | 12 | 10 | 3.75GB | 90sec | 5min 36sec |

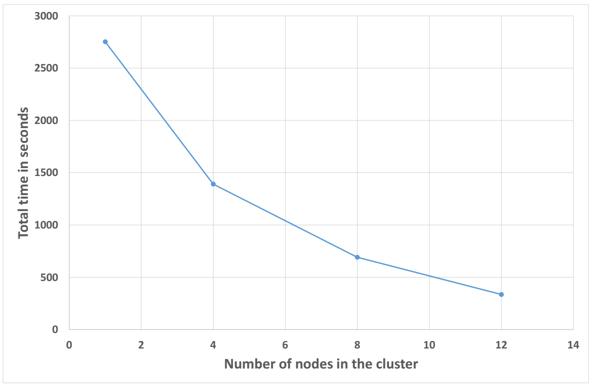

Figure 3. Total running time as a function of node numbers

for each cluster while Figure 3 indicates the running time as a function of node numbers. It takes about 46 minutes to index 6 GB data in one node cluster while it takes about 5.5 minutes in a twelve nodes cluster with 10 datanodes. The running time is almost linear to the number of nodes in the cluster since there is no dependency among the mappers. The reason why the performance is not perfectly linear is mostly due to the Hadoop setup overhead. Since we do not use reducers, there is no data shuffle phase. Table II shows Hadoop setup time is almost proportional to the size of the cluster. Current experiments show the scalability and efficiency of our system, and we can safely infer a scalable performance for bigger datasets in larger clusters based on the experimental results.

After the Lucene indices are built, we would like to test the performance of searching. Since Lucene relies on directory interface to make index searchable, there are two ways to search Lucene indices stored in HDFS. Boris implemented a memory-based approach where the indices were stored in memory for searching, which might not be suitable for large table sets [21]. The other approach is to copy the indices to local file system of each datanode. In this paper, we test the performance of querying the indices copied to one local file, which on average costs less than 1 second to get the results. Such response time is ideal for real-time data analytics; this also proves the efficiency of searching Lucene index.

## VI. Community discovery in Hurricane Sandy

There have been some work on finding communities in social network [22]–[24]. Most of researches in the field of community discoveries consider social connections, user mentions, description content etc., as the similarity measure for obtaining communities in the network [25]. Different clustering algorithms can thus be applied to group the people with a higher similarity into the same community, and those with less similarity into others. This works well for non-disaster situations because of steady social ties. However, when disaster strikes, retweet reflects the dynamic and temporal features as disaster develops.

In order to understand human reactions to natural disasters, we apply cluster algorithms to a similarity matrix based on retweets. Two people are considered to be connected when one person retweets the other. The similarity between these two is considered to be one. In order to construct the global similarity matrix, we have to find the shortest distance between any two connected pairs by retweets. Retweet graph captures the temporal feature of the disaster and reflects a certain level of trust. In this paper, we apply spectral clustering algorithm to the similarity matrix to find communities.

### A. Retweet Graph

Reweet is a way for a user to tweet content that has been posted by another user. The format is RT @username where username is the twitter name of the person being retweeted. As a result, we are only able to find the direct retweet relationship between two. In this paper, we define the distance of a retweeting pair as one, and the distance of non-retweeting relationship as infinity. Diagonals are defined as zero because a node has zero distance to itself.

For example, say user A retweets user B, and user B retweets user C, we can generate the retweet matrix in Table III. An issue with Table III is that the similarity of A and C stays infinity whereas it should be two. In order to calculate the similarities among all Twitter users, we implement Floyd's algorithm on the retweet matrix to generate the similarity matrix.

Table III
An illustration of retweet matrix describing the similarity among three Twitter users

| Similarity | A | B | C |
|---|---|---|---|
| A | 0 | 1 | $\infty$ |
| B | 1 | 0 | 1 |
| C | $\infty$ | 1 | 0 |

### B. Spectral Clustering

Spectral clustering is the technique of partitioning the rows of a matrix, similarity matrix in our case, according to their components in the top few singular vectors of the matrix. Spectral clustering techniques make use of the spectrum of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. There are different variants of spectral clustering implementation based on different types of graph Laplacian matrices. In our approach, we used the normalized spectral clustering according to Ng, Jordon, and Weiss [26] shown in Algorithm 1. The input of the algorithm includes the similarity matrix and the number K of clusters to construct. To build the similarity matrix, we implemented Floyd's algorithm to find shortest paths between all nodes. The output is K different communities.

**Algorithm 1** Normalized spectral clustering according to Ng, Jordan, and Weiss (2002)

```
Input: Retweeting pairs, number $k$ of clusters to constrcuct
Output: Cluster $A_1, \cdots A_k$ with $A_i = \{j | y_j \in C_i\}$
1: Construct a retweet matrix $S \in R^{n \times n}$ by using Floyd's algorithm to find the shortest path $S_{ij}$ if $i \neq j$, and $S_{ii} = 0$.
2: Form the similarity matrix $A \in R^{n \times n}$ defined by $A_{ij} = \exp(-S_{ij}^2 / 2\sigma^2)$.
3: Define $D$ to be the diagonal matrix whose $(i, i)$-element is the sum of $A$'s $i$-th row, and construct the Laplacian matrix $L = I - D^{-1/2} A D^{-1/2}$.
4: Compute the first $k$ eigenvectors $u_1, \cdots u_k$ of $L$ by running a Singular Value Decomposition (SVD) on $L$.
5: Let $U$ be the matrix containing the vectors $u_1, \cdots u_k$ as columns.
6: Form the matrix $T \in R^{n \times k}$ from $U$ by normalizing the rows to norm 1, that is set $t_{ij} = u_{ij} / (\sum_k u_{ik}^2)^{1/2}$.
7: For $i = 1, \cdots, n$, let $y_i \in R^k$ be the vector corresponding to the $i$-th row of $T$.
8: Cluster the points $(y_i)_{i=1,\cdots,n}$ with the $k$-means algorithm into clusters $C_1, \cdots, C_k$.
```

### C. Experimental Results

To retrieve Hurricane Sandy related tweets in NYC from our *Twitter* table, we search the Lucene indices using a keyword based approach by specifying the date to be October

26-31, 2012, location New York City, and keywords like "Hurricane", "Sandy", "Evacuation", and "Emergency" etc. After removing duplicates, the retweets in the result data set are stored in *SandyTwitter* table.

Because the storm hit NYC on October 29, 2012, we divided the tweets into three groups: October 26-28 before landing on NYC, October 29 landing on NYC, and October 30-31 after landing. Table IV lists the statistics of Twitter data in NYC from October 26 to October 31. As we can see there is a slightly decrease in the number of tweets and Twitter users on October 29 as Hurricane Sandy impacted NYC. And the number almost doubled afterwards. Another observation is that more than a quarter of tweets are retweets. For each timeline, we tentatively clustered the data into 5 communities. We ended up with 15 communities. We also defined the influential user by counting the number of retweets; user with a higher number of retweets was considered to have a bigger impact in the community. Figure 4, generated by Gephi [27], shows the members of one community on October 29 where *NYCMayorsOffice*, *NYGovCuomo*, *MikeBloomberg*, *AHurricaneSandy*, *sethmeyers21*, *nydailynews*, *NYTMetro*, and *NewYorkPost* could be considered as the influential people.

Table IV
New York City tweets statistics during Hurricane Sandy from October 26 to October 31, 2012

| NYC | Oct 26-28 | Oct 29 | Oct 30-31 |
|---|---|---|---|
| Retweets/Tweets | 1181/4551 | 945/3651 | 2393/8027 |
| Percentage | 26% | 25.8% | 29.8% |
| Number of users | 1590 | 1375 | 3071 |

In addition, we utilize Latent Semantic Indexing [28] (LSI) to identify the topics for each community on different days. We first generate a term-document matrix based on the tweet content from each community. The term-document matrix, which is a bag-of-words representation, is later transformed into a TF-IDF model. At last we transform this TF-IDF model into a latent space of a lower dimensionality. In the reduced space, we select top 10 topics, which are composed of individual keywords.

Figure 5 shows four word clouds generated by keywords from different communities before, during, and after Hurricane Sandy hit NYC. The top one shows the topics of a community before Hurricane Sandy hit where people are concerned about the approaching of Hurricane Sandy, and they are talking about *alerts, evacuation, and leave*. The middle two word clouds are from two different communities on October 29, 2012 when Hurricane Sandy hit NYC. One mainly talked about the closing of Battery Tunnel and Holland Tunnel as evidenced by the following tweet posted by *NYCMayorsOffice: Hugh Carey Brooklyn Battery Tunnel*

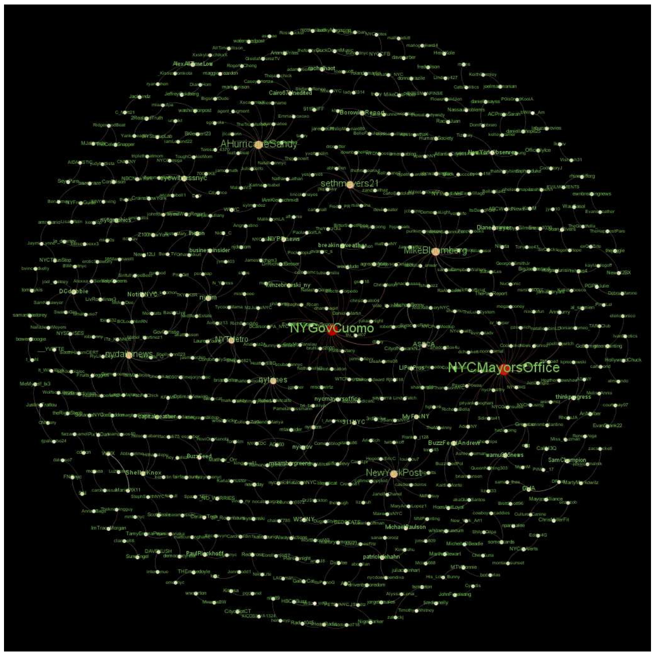

Figure 4. A graph generated by Gephi showing all members in one community. Different nodes represent different Twitter users. Nodes are connected if one retweets the other. The size and color of the node is proportional to the number of retweets the node gets.

*Holland Tunnel closing PM today per Sandy*. The other more focused on emergency situations. Frankenstorm refers to Hurricane Sandy. The last one shows when the Hurricane is gone, most people are tweeting about *thanking NYPD, saving lives, and restoring water*.

## VII. Discussion

This data analytics system is scalable in the sense that data are distributed and indexed in HDFS. However, there is still room for improving the scalability in terms of parallelizing spectral clustering algorithm; future work can utilize MapReduce to implement the spectral clustering algorithm. Our implementation consists of three parts: 1. Floyd's algorithm to find shortest paths for all pairs to build similarity matrix 2. Singular Value Decomposition (SVD) of the similarity matrix 3. K-means clustering applied to a lower dimension data space built from step 2.

In our experiment, it took more than 50 minutes to cluster 3000+ points in a laptop with Intel(R) Core(TM) i5-2520M CPU, 4.0GB RAM, 64-bit Windows 7 and Oracle Java 7 where Floyd's algorithm takes 40 seconds, SVD takes 45 minutes, and k-means takes about 1 minute. The parallelization and optimization comes down to how to use MapReduce to implement SVD. Reza computed the singular values and the right singular vectors of a matrix in a MapReduce environment [29]. Mohamed Hefeeda etc. designed a distributed approximate spetral clustering (DASC) algorithm,

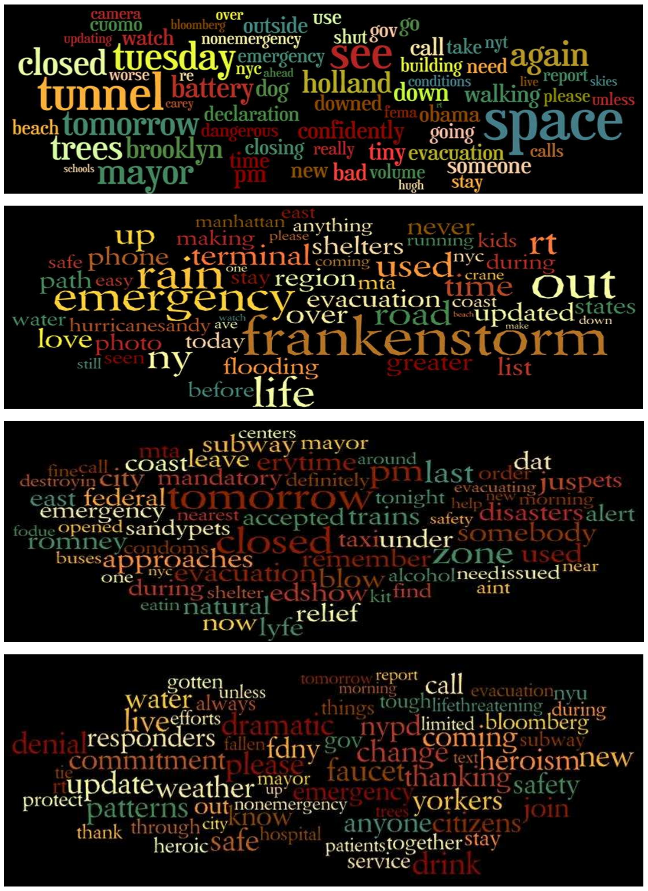

Figure 5. Word clouds generated from different communities according to the timeline. The first is before Sandy hit NYC, the second and third are two different communities when Sandy hit NYC and the fourth is after Sandy left NYC.

and implemented DASC in the MapReduce framework [30]. Moreover, in this work we search the Lucene indices by copying them back to one index directory stored locally; this becomes impossible as the size of index grows. We plan to distribute queries to each datanode where index is located and combine the results using MapReduce.

## VIII. Conclusion

This paper focuses on a social media data analytics system for community discovery in Hurricane Sandy Twitter based on HBase and Lucene. Experiments demonstrate the scalability and efficiency for analyzing Twitter data and building Lucene index for answering multiple dimensional queries. This system takes advantage of Hadoop in many aspects. Firstly, we use HBase and HDFS as the data center to overcome the scalability and large volume data challenges brought by social media data. Secondly, we rely on MapReduce to build a Lucene index to empower HBase with the full-text searching ability and utilize parallelization to overcome computing complexity and big data processing.

In addition, we develop a novel approach to discover communities based on retweet matrix using spectral clustering algorithm. As the disaster unfolds, influential users are identified, topical changes are observed, and the community evolvement is demonstrated. This contributes to a finer level of understanding human reactions in the face of natural disasters in social media.

## Acknowledgment

Yin Huang is supported by IBM CAS fellowship. We would like to thank Craig Statchuk for helpful advice.

## References

[1] Kaigo, Muneo. “Social media usage during disasters and social capital: Twitter and the Great East Japan earthquake.” Keio Communication Review 34 (2012): 19-35.

[2] Han Dong, Milton Halem, and Shujia Zhou. “Social media data analytics applied to Hurricane Sandy.” In Social Computing (SocialCom), 2013 International Conference on, pp. 963-966. IEEE, 2013.

[3] Han, Jing, E. Haihong, Guan Le, and Jian Du. “Survey on NoSQL database.” In Pervasive computing and applications (ICPCA), 2011 6th international conference on, pp. 363-366. IEEE, 2011.

[4] Li, Ning, Jun Rao, Eugene Shekita, and Sandeep Tata. “Leveraging a scalable row store to build a distributed text index.” In Proceedings of the first international workshop on Cloud data management, pp. 29-36. ACM, 2009.

[5] Hatcher, Erik, Otis Gospodnetic, and Michael McCandless. “Lucene in action.” (2004).

[6] Nutch, https://nutch.apache.org/

[7] Solr, https://lucene.apache.org/solr/

[8] ElasticSearch, http://www.elasticsearch.org/

[9] Wang, Xufei, Lei Tang, Huiji Gao, and Huan Liu. “Discovering overlapping groups in social media.” In Data Mining (ICDM), 2010 IEEE 10th International Conference on, pp. 569-578. IEEE, 2010.

[10] Apache Storm, http://storm.incubator.apache.org/

[11] HBasene project, https://github.com/akkumar/hbasene/

[12] lily, http://www.lilyproject.org/

[13] IHBase, https://github.com/ykulbak/ihbase/.

[14] Xiaoming Gao, Vaibhav Nachankar, and Judy Qiu. “Experimenting Lucene Index on HBase in an HPC Environment.” HPCDB ’11 Proceedings of the first annual workshop on high performance computing meets databases, pp. 25-28. ACM, 2011

[15] Bron C, Kerbosch J. “Algorithm 457: finding all cliques of an undirected graph.” Commun ACM 16(9):575-577, 1973

[16] Newman MEJ. “Fast algorithm for detecting community structure in networks.” Phys Rev E69:066133, 2004a

[17] Girvan M, Newman MEJ. “Community structure in social and biological networks.” Proc Natl Acad Sci USA 99(12):78217826, 2002

[18] Van Dongen S. “Graph clustering by flow simulation.” Ph.D. Thesis, Dutch National Research Institute for Mathematics and Computer Science, Utrecht, Netherlands, 2000

[19] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. “Community detection in Social Media.” Data Mining and Knowledge Discovery, May 2012, Volume 24, Issue 3, pp 515-554.

[20] Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. “Bigtable: A distributed storage system for structured data.” ACM Transactions on Computer Systems (TOCS) 26, no. 2 (2008): 4.

[21] Boris Lublinsky and Mike Segel. http://www.infoq.com/articles/LuceneHbase

[22] S. Wasserman and K. Faust. “Social Network Analysis”. Cambridge University Press, Cambridge (1994)

[23] J. Scott. “Social Network Analysis: A Handbook”. Sage Publications, London, 2nd edition (2000).

[24] D. J. Watts and S. H. Strogatz. “Collective dynamics of ‘small-world’ networks”. Nature 393, 440-442 (1998).

[25] Pulkit Goyal, http://pulkitgoyal.in/2012/03/17/similarity-metrics-twitter/

[26] Ng, Andrew Y., Michael I. Jordan, and Yair Weiss. “On spectral clustering: Analysis and an algorithm.” Advances in neural information processing systems 2 (2002): 849-856.

[27] Bastian, Mathieu, Sebastien Heymann, and Mathieu Jacomy. “Gephi: an open source software for exploring and manipulating networks.” In ICWSM. 2009.

[28] Deerwester, Scott C., Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. “Indexing by latent semantic analysis.” JASIS 41, no. 6 (1990): 391-407.

[29] Reza Bosagh Zadeh and Gunnar Carlsson. “Dimension Independent Matrix Square using MapReduce.” CoRR journal, volume abs/1304.1467, 2013.

[30] Mohamed Hefeeda, Fei Gao, and Wael Abd-Almageed. “Distributed approximate spectral clustering for large-scale datasets.” HPDC’12 Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing, pp. 223-234. ACM 2012.